

Improving Code Smell Predictions in Continuous Integration by Differentiating Organic from Cumulative Measures

Abdullah Mamun¹, Mirosław Staron¹, Christian Berger¹,
Regina Hebig¹, and Jörgen Hansson²

¹ Department of Computer Science and Engineering, Chalmers | University of Gothenburg, Gothenburg, Sweden. abdullah.mamun@chalmers.se,
[\[mirosław.staron, christian.berger, regina.hebig\]@cse.gu.se](mailto:[mirosław.staron, christian.berger, regina.hebig]@cse.gu.se)

² School of Informatics, University of Skövde, Sweden, jorgen.hansson@his.se

A Appendix

Code smells used in this study are listed in the following Table.

"assert" should only be used with boolean variables
"clone" should not be overridden
"entrySet()" should be iterated when both the key and value are needed
"finalize" should not set fields to "null"
"FIXME" tags should be handled
"for" loop incrementers should modify the variable being tested in the loop's stop condition
"for" loop stop conditions should be invariant
"indexOf" checks should not be for positive numbers
"indexOf" checks should use a start position
"instanceof" operators that always return "true" or "false" should be removed
"Lock" objects should not be "synchronized"
"Object.finalize()" should remain protected (versus public) when overriding
"Object.wait(...)" should never be called on objects that implement <code>java.util.concurrent.locks.Condition</code>
"private" methods called only by inner classes should be moved to those classes
"readObject" should not be "synchronized"
"readResolve" methods should be inheritable
"ResultSet.isLast()" should not be used
"static" members should be accessed statically
"StringBuilder" and "StringBuffer" should not be instantiated with a character
"switch" statements should end with "default" clauses
"switch" statements should have at least 3 "case" clauses
"switch" statements should not contain non-case labels
"Thread.sleep" should not be used in tests
"Threads" should not be used where "Runnables" are expected
"toString()" should never be called on a String object

"URL.hashCode" and "URL.equals" should be avoided
A "while" loop should be used instead of a "for" loop
A field should not duplicate the name of its containing class
Array designators "[]" should be located after the type in method signatures
Array designators "[]" should be on the type, not the variable
Assignments should not be made from within sub-expressions
Boolean checks should not be inverted
Boolean literals should not be redundant
Boxing and unboxing should not be immediately reversed
Branches should have sufficient coverage by unit tests
Case insensitive string comparisons should be made without intermediate upper or lower casing
Catches should be combined
Child class fields should not shadow parent class fields
Child class methods named for parent class methods should be overrides
Class names should comply with a naming convention
Class names should not shadow interfaces or superclasses
Class variable fields should not have public accessibility
Classes from "sun.*" packages should not be used
Classes named like "Exception" should extend "Exception" or a subclass
Classes should not be empty
Classes that override "clone" should be "Cloneable" and call "super.clone()"
Classes with only "static" methods should not be instantiated
Collapsible "if" statements should be merged
Collection.isEmpty() should be used to test for emptiness
Collections.emptyList(), emptyMap() and emptySet() should be used instead of Collections.EMPTY_LIST, EMPTY_MAP and EMPTY_SET
Constant names should comply with a naming convention
Constants should not be defined in interfaces
Constructor injection should be used instead of field injection
Dead stores should be removed
Declarations should use Java collection interfaces such as "List" rather than specific implementation classes such as "LinkedList"
Dependencies should not have "system" scope
Deprecated "\${pom}" properties should not be used
Deprecated elements should have both the annotation and the Javadoc tag
Double Brace Initialization should not be used
Empty arrays and collections should be returned instead of null
Empty statements should be removed
Enumeration should not be implemented
Exception types should not be tested using "instanceof" in catch blocks
Exit methods should not be called
Expressions should not be too complex
Field names should comply with a naming convention

Fields in non-serializable classes should not be "transient"
Future keywords should not be used as names
Generic wildcard types should not be used in return parameters
Inner class calls to super class methods should be unambiguous
Interface names should comply with a naming convention
JUnit rules should be used
Labels should not be used
Local variable and method parameter names should comply with a naming convention
Local Variables should not be declared and then immediately returned or thrown
Local variables should not shadow class fields
Loops should not contain more than a single "break" or "continue" statement
Maps with keys that are enum values should be replaced with EnumMap
Method names should comply with a naming convention
Method overrides should not change contracts
Methods and field names should not be the same or differ only by capitalization
Methods named "equals" should override Object.equals(Object)
Methods of "Random" that return floating point values should not be used in random integer generation
Methods should not be empty
Methods should not have too many parameters
Methods should not return constants
Modifiers should be declared in the correct order
Multiple variables should not be declared on the same line
Nested "enum"s should not be declared static
Nested blocks of code should not be left empty
Nested code blocks should not be used
Non-constructor methods should not have the same name as the enclosing class
Null should not be returned from a "Boolean" method
Objects should not be created only to "getClass"
Only static class initializers should be used
Overriding methods should do more than simply call the same method in the super class
Package declaration should match source file directory
Package names should comply with a naming convention
Parsing should be used to convert "Strings" to primitives
Primitive wrappers should not be instantiated only for "toString" or "compareTo" calls
Primitives should not be boxed just for "String" conversion
Public constants and fields initialized at declaration should be "static final" rather than merely "final"
Public methods should throw at most one checked exception
Public types, methods and fields (API) should be documented with Javadoc
Redundant casts should not be used
Return of boolean expressions should not be wrapped into an "if-then-else" statement
Silly bit operations should not be performed

Silly math should not be performed
Standard outputs should not be used directly to log anything
Statements should be on separate lines
Static non-final field names should comply with a naming convention
String function use should be optimized for single characters
String.valueOf() should not be appended to a String
Strings literals should be placed on the left side when checking for equality
Strings should not be concatenated using '+' in a loop
Subclasses that add fields should override "equals"
Switch cases should end with an unconditional "break" statement
TestCases should contain tests
Tests should not be ignored
The default unnamed package should not be used
The diamond operator (";") should be used
The members of an interface declaration or class should appear in a pre-defined order
The signature of "finalize()" should match that of "Object.finalize()"
Throws declarations should not be superfluous
Try-with-resources should be used
Type parameter names should comply with a naming convention
Unused "private" fields should be removed
Unused "private" methods should be removed
Unused labels should be removed
Unused local variables should be removed
Unused method parameters should be removed
Unused type parameters should be removed
Useless "if(true) ..." and "if(false)..." blocks should be removed
Useless imports should be removed
Useless parentheses around expressions should be removed to prevent any misunderstanding
Utility classes should not have public constructors
"@Deprecated" code should not be used
"@Override" annotation should be used on any method overriding (since Java 5) or implementing (since Java 6) another one

Table 1: Code smells used in this study.